axioms

Article

An Approximation Algorithm for a Variant of Dominating Set Problem

Limin Wang [1,*] and Wenqi Wang [2]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2] School of Mathematical Science & Institute of Mathematics, Nanjing Normal University, Nanjing 210023, China; wenqiwangcc@nnu.edu.cn
* Correspondence: dg1533032@smail.nju.edu.cn or wanglimin0721@163.com

Abstract: In this paper, we consider a variant of dominating set problem, i.e., the total dominating set problem. Given an undirected graph $G = (V, E)$, a subset of vertices $T \subseteq V$ is called a total dominating set if every vertex in V is adjacent to at least one vertex in T. Based on LP relaxation techniques, this paper gives a distributed approximation algorithm for the total dominating set problem in general graphs. The presented algorithm obtains a fractional total dominating set that is, at most, $k(1 + \Delta^{\frac{1}{k}})\Delta^{\frac{1}{k}}$ times the size of the optimal solution to this problem, where k is a positive integer and Δ is the maximum degree of G. The running time of this algorithm is constant communication rounds under the assumption of a synchronous communication model.

Keywords: distributed algorithm; approximation; domination; communication rounds

MSC: 05C90

check for updates

Citation: Wang, L.; Wang, W. An Approximation Algorithm for a Variant of Dominating Set Problem. *Axioms* **2023**, *12*, 506. https://doi.org/10.3390/axioms12060506

Academic Editor: Mircea Merca

Received: 12 March 2023
Revised: 19 May 2023
Accepted: 20 May 2023
Published: 23 May 2023

1. Introduction

A subset $D \subseteq V$ of vertices is called a *dominating set* of G if every vertex in $V - D$ is adjacent to at least one vertex in D. Domination is one of the central problems in the theoretical study of network design and is a widely used class of combinatorial optimization problems with applications in surveillance communications [1], coding theory [2], cryptography [3,4], complex ecosystems [5], electric power networks [6], etc. More practical applications and theoretical results can be found in [7–10].

A dominating set can be used to optimize the number and location of servers in a network. It is also used in some routing protocols for ad hoc networks [11,12], when it satisfies certain specific properties (weakly connected or connected). In a dominating set, the vertices represent the servers in a network, which can provide essential services to the users. However, when a server is attacked or suddenly crashes, the service provided will be affected. As each server is adjacent to another server in a total dominating set, it can provide greater fault tolerance and serve as a backup when a server crashes. Because each vertex is dominated by another vertex in the total dominating set, this special domination property is precisely adapted to peer-to-peer networks [13]. A total dominating set is a variant of dominating sets; Cockayne introduced its definition in [14]. In a network graph $G = (V, E)$, a subset $T \subseteq V$ of vertices of G is called a *total dominating set* (TDS) if each vertex in V is adjacent to at least one vertex in T. For a TDS T of G, if no proper subset of T is a TDS of G, T is called minimal. The minimum total dominating set (MTDS) problem is to find a total dominating set of the minimum size.

In this paper, we present a distributed approximation algorithm for this MTDS problem based on LP relaxation techniques. The algorithm obtains a fractional total dominating set that is, at most, $k(1 + \Delta^{\frac{1}{k}})\Delta^{\frac{1}{k}}$ times the size of the optimal solution for the TDS problem in general graphs, where k is a positive integer and Δ is the maximum degree. The key to designing this algorithm is to use the maximum dynamic degree in the second neighborhood of the vertex, where the *dynamic degree* is the number of white vertices in the open neighborhood of a vertex at a given time. The communication model used in this paper is synchronous [15]. That is to say, in each communication round, every vertex can send a message to each of its neighbors in the given graph; the time complexity of the algorithm is to compute the number of communication rounds. In this paper, the number of communication rounds for our algorithm is a constant. Unlike the existing distributed algorithms, the algorithms in [16,17] focus on special graphs (planar graphs without three or four cycles), while this paper is concerned with general graphs. The self-stabilizing distributed algorithm designed in [13] can obtain a minimal total dominating set in polynomial time $O(mn)$ (m is the number of edges in the graph and n is the number of vertices in the graph) in general graphs. Comparatively, the distributed algorithm designed in this paper, under a synchronous communication model, can obtain a non-trivial approximation ratio in general graphs with a constant number of rounds. The time complexity of our algorithm is $O(k^2)$, where k is a positive integer.

The paper is organized as follows. We first give some complexity and algorithmic results on the TDS problem in Section 2. Section 3 introduces some necessary notations and presents the linear programming relaxation of the TDS problem. In Section 4, based on LP relaxation techniques and the definition of the maximum dynamic degree in the second neighborhood of the vertex, we design a distributed approximation algorithm for the TDS problem. In Section 5, we summarize this paper and introduce some future work.

2. Related Work

The concept of total domination in graphs was introduced in [14], and has been extensively studied in [18–21]. Garey et al. showed that the problem of finding an MTDS is NP-hard for general graphs [22] and is also MAX SNP-hard [23–25]. Laskar et al. [26] gave a linear time algorithm to compute the total domination number in a tree and showed that it is NP-complete to find an MTDS in undirected path graphs.

Henning et al. [27] proposed a heuristic algorithm to find a TDS whose size is at most $n(1 + \ln \delta)/\ln \delta$, where n is the number of vertices in G and $\delta \geq 2$ is the minimum degree. An $(H_\Delta - \frac{1}{2})$-approximation algorithm was presented in [28], where $H_i = \sum_{h=1}^{i} \frac{1}{h}$ is the i-th harmonic number. In [28], they also showed that there exist two constants, $a \geq 3$ and $b > 0$, such that, for each $\Delta \geq a$, it is NP-hard to approximate MTDS within factor $\ln \Delta - b \ln \ln \Delta$ in bipartite graphs. Zhu [29] designed a greedy algorithm for the TDS problem; the performance ratio of the algorithm, $\ln(\Delta - 0.5) + 1.5$, showed that it is NP-hard to approximate MTDS within factor $(1 - \varepsilon) \ln |V|$ ($\varepsilon > 0$) in sparse graphs, unless $NP \subseteq DTIME(|V|^{O(\log \log |V|)})$, within factor 391/390 for three-bounded degree graphs, within factor 681/680 for three-regular graphs, and within factor 250/249 for four-regular graphs. More complexity results can be found in [20].

Schaudt et al. [30] showed that it is NP-hard to seek for a TDS T such that the subgraph $G[T]$ belongs to a member of a given graph class, such as perfect graphs, bipartite graphs, asteroidal-triple-free graphs, interval graphs, unicyclic graphs, etc. By integrating a genetic algorithm and local search, Yuan et al. [31] designed a hybrid evolutionary algorithm for the TDS problem.

In a planar graph without three cycles, a 16-approximation algorithm was presented in [16] for the TDS problem. Alipour and Jafari [17] presented a 9-approximation algorithm for the TDS problem in a planar graph without four cycles. Bahadir [32] provided an algorithm to determine whether a graph satisfies $\gamma_t(G) = 2\gamma(G)$ or not, where $\gamma_t(G)$ is the total domination number and $\gamma(G)$ is the domination number of a graph G. Hu et al. [33] designed an improved local search framework to deal with the TDS problem by analyzing the algorithm in [31]. In [34], Jana and Das first showed that it is NP-complete to find an MTDS in a given geometric unit disk graph. Next, they presented an 8-approximation algorithm for the TDS problem in the geometric unit disk graphs, with the running time of the algorithm as $O(n \log k)$, where k is the size of the output result of their algorithm.

By using the shifting strategy technique in [35], a polynomial time approximation scheme (PTAS) was given in [34] for the TDS problem in the geometric unit disk graphs.

In a network, a scheduler is called fair if a continuously enabled node will eventually be selected by the scheduler. Otherwise, an unfair scheduler can only guarantee the progress of the global system [13]. For the minimal TDS problem, Goddard et al. [36] presented a self-stabilizing algorithm whose running time is exponential under the unfair central scheduler, i.e., only one enabled node performs a move step at a time. Belhoul et al. [13] designed another self-stabilizing algorithm under the unfair distributed scheduler (any non-empty subset of enabled nodes will be selected to perform the move step); however, the running time of its algorithm is polynomial time.

3. Preliminaries

Firstly, we introduce some notations. Given a graph $G = (V, E)$, for the sake of discussion, we assume that $V = \{v_1, v_2, \ldots, v_n\}$. For a vertex $v_i \in V$, let $N(v_i) = \{v_j \in V | v_i v_j \in E\}$ denote the open neighborhood of v_i. Denote the degree of v_i in graph by $\delta(v_i)$, which is the number of vertices in the neighborhood $N(v_i)$, i.e., $\delta(v_i) = |N(v_i)|$. Let Δ denote the maximum degree of all vertices in G. Denote the maximum degree of v_i in $N(v_i)$ by $\delta^{(1)}(v_i) := \max_{v_j \in N(v_i)} \delta(v_j)$, and the maximum degree of v_i in the second neighborhood by $\delta^{(2)}(v_i) := \max_{v_j \in N(v_i)} \delta^{(1)}(v_j)$.

Secondly, we give the linear programming relaxation for the total dominating set problem. Given an undirected graph $G = (V, E)$, $T \subseteq V$ is a total dominating set of G if each vertex $v \in V$ satisfies $N(v_i) \bigcap T \neq \emptyset$. For each vertex $v_i \in V$, we assign a corresponding binary variable x_i. If x_i is set to 1 if and only if v_i is a member of the total dominating set T, i.e., $v_i \in T$. Hence, T is called a total dominating set of G if, and only if, it satisfies $\sum_{v_j \in N(v_i)} x_j \geq 1$ for every vertex $v_i \in V$. Denote the adjacency matrix for G by N. The MTDS problem can be formulated as the following integer linear programming (ILP_{MTDS}):

$$\begin{aligned} \textbf{minimize} \quad & \sum_{i=1}^{n} x_i \\ \textbf{s.t.} \quad & N \cdot \overline{x} \geq \overline{1} \\ & \overline{x} \in \{0,1\}^n, \end{aligned}$$

where the first constraint indicates that if the vertex v_i satisfies $\sum_{v_j \in N(v_i)} x_j \geq 1$, then the vertex v_i is a member of the total dominating set.

By relaxing the constraints, we can obtain the linear programming relaxation (LP_{MTDS}) of ILP_{MTDS}:

$$\begin{aligned} \textbf{minimize} \quad & \sum_{i=1}^{n} x_i \\ \textbf{s.t.} \quad & N \cdot \overline{x} \geq \overline{1} \\ & \overline{x} \geq \overline{0}. \end{aligned}$$

The dual programming (DLP_{MTDS}) of LP_{MTDS} is

$$\begin{aligned} \textbf{maximize} \quad & \sum_{i=1}^{n} y_i \\ \textbf{s.t.} \quad & N \cdot \overline{y} \leq \overline{1} \\ & \overline{y} \geq \overline{0}, \end{aligned}$$

where the first constraint indicates that if we introduce a positive value y_i for every vertex v_i, then $\sum y_i$ of all vertices in $N(v_i)$ of every vertex v_i is at most 1. This means that the sum of the corresponding x_i in $N(v_i)$ of every vertex v_i is at least 1.

4. Algorithm for the TDS Problem

In this section, based on the LP relaxation techniques and the definition of the maximum dynamic degree in the second neighborhood of the vertex, we present a distributed approximation algorithm for the TDS problem by integrating the techniques in [37,38].

Similarly to the assumption about dominating sets in [39,40], for each vertex v_i, if the vertex satisfies $\sum_{v_j \in N(v_i)} x_j \geq 1$, we say that the vertex is covered, and it will be colored black. Initially all vertices are colored white. Denote the *dynamic degree* of a vertex v_i by $\tilde{\delta}(v_i)$, and use it to compute the number of white vertices in $N(v_i)$. Hence, when the algorithm starts, we have $\tilde{\delta}(v_i) = \delta(v_i)$.

Before analyzing our algorithm, we give a brief description of the algorithm. We introduce a variable x_i to every vertex v_i, x_i is initialized as 0, as the algorithm runs, it gradually increases. The outer loop iteration $\hbar$ is mainly to reduce $\tilde{\delta}(v_i)$. In the outer loop iteration of the algorithm, let $\tau^{(1)}(v_i)$ and $\tau^{(2)}(v_i)$ denote the maximum dynamic degree in $N(v_i)$ and the second neighborhood of v_i, respectively. In each inner loop iteration, only the vertices that satisfy $\tilde{\delta}(v_i) \geq \tau^{(2)}(v_i)^{\frac{\hbar}{\hbar+1}}$ increase the corresponding x_i. We call these vertices *active*. For a white vertex v_i, denote the number of active vertices in $N(v_i)$ by $a(v_i)$. If the vertex v_i is colored black, let $a(v_i) = 0$. For each vertex $v_j \in N(v_i)$, let $a^{(1)}(v_i)$ denote the maximum $a(v_j)$. The key to solving the total dominating set problem in this paper is how to use the maximum dynamic degree in the second neighborhood of the vertex to design Algorithm 1.

First, we illustrate the execution of Algorithm 1 with an example in Figure 1, where $k = 5$ and $\Delta = 5$. Figure 1a is the topology graph of the network with $n = 12$, the $\tilde{\delta}(v_i)$ of each vertex is marked on the graph, initial values of variables x_i, $\tilde{\delta}(v_i)$, $\tau^{(2)}(v_i)$ are shown in Table 1. In Figure 1b, when $\hbar = k - 1 = 4$ and $m = 4$, v_6, v_8 satisfy $\tilde{\delta}(v_i) \geq \tau^{(2)}(v_i)^{\frac{\hbar}{\hbar+1}}$, v_6, v_8 are active vertices, hence v_6, v_8 are selected, then the corresponding x_6, x_8 of v_6, v_8 are changed from 0 to $a^{(1)}(v_i)^{-\frac{m}{m+1}} \approx 0.57$. According to lines 20–22 of Algorithm 1, v_7 satisfies $\sum_{v_j \in N(v_7)} x_j \geq 1$, hence v_7 is colored black and we need to update $\tilde{\delta}(v_i)$. New values of variables $\tau^{(2)}(v_i)^{\frac{\hbar}{\hbar+1}}$, $a(v_i)$, $a^{(1)}(v_i)$, x_i, $\tilde{\delta}(v_i)$ are shown in Table 1. When $m = 3$, new values of variables $a(v_i)$, $a^{(1)}(v_i)$, x_i, $\tilde{\delta}(v_i)$ are shown in Table 1, we continue to execute the algorithm, and v_4, v_5, v_{11}, v_{12} are colored black in Figure 1c. When $m : 2 \rightarrow 0$, the dynamic degrees $\tilde{\delta}(v_i)$ of all vertices are less than the corrsponding $\tau^{(2)}(v_i)^{\frac{4}{5}}$, so the outer iteration $\hbar = 4$ ends, and we update $\tau^{(2)}(v_i)$ in Table 1. By continuing to execute Algorithm 1, when $\hbar : 3 \rightarrow 0$ and $m : 4 \rightarrow 0$, the coloring process of all vertices and the changes in $\tilde{\delta}(v_i)$ are shown in Figure 1d–f. Initially, all $x_i = 0$. By executing the algorithm, the final x_i of all vertices is shown in Figure 1g. So, we can determine that the size of the output solution of Algorithm 1 is $\sum_{i=1}^{12} x_i = 4 + 0.57 = 4.57$.

The optimal TDS for this example is shown in Figure 2. Hence, we can see that the optimal TDS in this case is $\{v_4, v_6, v_8, v_9\}$, so its size is 4. The performance ratio of Algorithm 1 is $\frac{4.57}{4} \approx 1.14$ in this case.

Second, we give some lemmas that will be used to analyze the approximation ratio of Algorithm 1.

Algorithm 1 Approximating LP_{MTDS}

Input: Given a graph $G = (V, E)$, a positive integer k.
Output: A feasible solution $\overline{x}$ for LP_{MTDS}.

initially $x_i := 0$, $\tilde{\delta}(v_i) := \delta(v_i)$, $V' := V$;
calculate $\delta^{(2)}(v_i)$, set $\tau^{(2)}(v_i) := \delta^{(2)}(v_i)$;
for $\hbar := k-1$ to 0 by -1 **do**
  $(\star \quad \tilde{\delta}(v_i)$, set $z_i := 0 \quad \star)$
  for $m := k-1$ to 0 by -1 **do**
    if $\tilde{\delta}(v_i) \geq \tau^{(2)}(v_i)^{\frac{\hbar}{\hbar+1}}$, $v_i \in V'$ **then**
      send 'active vertex' to all neighbors
    end if
    calculate $a(v_i) := |\{v_j \in N(v_i) | \textit{the vertex } v_j \textit{ is active vertex}\}|$
    if the vertex v_i is colored 'black' **then**
      $a(v_i) := 0$
    end if
    send $a(v_i)$ to all vertices in $N(v_i)$;
    calculate $a^{(1)}(v_i) := \max_{j \in N(v_i)}\{a(v_j)\}$;
    $(\star \quad a(v_i), a^{(1)}(v_i) \quad \star)$
    if $\tilde{\delta}(v_i) \geq \tau^{(2)}(v_i)^{\frac{\hbar}{\hbar+1}}$, $v_i \in V'$ **then**
      $x_i := \max\{x_i, a^{(1)}(v_i)^{-\frac{m}{m+1}}\}$
    end if
    send x_i to all vertices in $N(v_i)$;
    if $\sum_{v_j \in N(v_i)} x_j \geq 1$ **then**
      the vertex v_i is colored 'black'
    end if
    send the color of vertex v_i to all vertices in $N(v_i)$;
    update $\tilde{\delta}(v_i) := |\{v_j \in N(v_i) | \textit{the vertex } v_j \textit{ is white}\}|$
  end for
  $(\star \quad z_i \quad \star)$
  send $\tilde{\delta}(v_i)$ to all vertices in $N(v_i)$;
  calculate $\tau^{(1)}(v_i) := \max_{j \in N(v_i)}\{\tilde{\delta}(v_j)\}$;
  send $\tau^{(1)}(v_i)$ to all vertices in $N(v_i)$;
  update $\tau^{(2)}(v_i) := \max_{j \in N(v_i)}\{\tau^{(1)}(v_j)\}$;
  if $\tilde{\delta}(v_i) = \tau^{(2)}(v_i) = 0$ **then**
    $V' =: V' - \{v_i\}$
  end if
  if $V' \neq \emptyset$ **then**
    continue to execute the algorithm
  end if
end for

Lemma 1. *When each outer loop iteration $\hbar$ starts, for each vertex $v_i \in V$, we can obtain*

$$\tilde{\delta}(v_i) \leq \Delta^{\frac{\hbar+1}{k}}.$$

Proof of Lemma 1. We prove it by using induction.

Firstly, when $\hbar = k-1$, the condition $\Delta^{\frac{\hbar+1}{k}} = \Delta$. Due to $\tilde{\delta}(v_i)$ being the dynamic degree in $N(v_i)$ and Δ being the maximum degree, it is clear that $\tilde{\delta}(v_i) \leq \Delta$.

Secondly, in order to prove that the other iterations also hold, we use the algorithm in [37] to approximate LP_{MTDS} (the modified algorithm will be available in Appendix A), where all vertices know Δ in their algorithm. Thus, similar to their algorithmic analysis, in the last step ($m = 0$) of the preceding outer loop iteration $\hbar + 1$, the x_i of all vertices with $\tilde{\delta}(v_i) \geq \Delta^{\frac{\hbar+1}{k}}$ is changed to 1. Hence, all vertices in $N(v_i)$ will be colored black, so $\tilde{\delta}(v_i)$ will be changed to 0. Thus, the dynamic degrees of all vertices with $\tilde{\delta}(v_i) \geq \Delta^{\frac{\hbar+1}{k}}$ is changed to 0, and the dynamic degrees of the other vertices clearly satisfy this inequality $\tilde{\delta}(v_i) \leq \Delta^{\frac{\hbar+1}{k}}$. Therefore, by using the algorithm in [37], we can verify that $\tilde{\delta}(v_i) \leq \Delta^{\frac{\hbar+1}{k}}$ holds when each outer loop iteration $\hbar$ starts. Hence, for our algorithm, we only need to show that the x_i of all vertices with $\tilde{\delta}(v_i) \geq \Delta^{\frac{\hbar}{k}}$ is changed to 1 when each inner loop iteration ends ($m = 0$). According to lines 17–19 of Algorithm 1, we can see that the x_i of all vertices with $\tilde{\delta}(v_i) \geq \tau^{(2)}(v_i)^{\frac{\hbar}{\hbar+1}}$ are changed to 1 when $m = 0$. Therefore, we only need to prove that $\tau^{(2)}(v_i)^{\frac{\hbar}{\hbar+1}} \leq \Delta^{\frac{\hbar}{k}}$ for each vertex v_i.

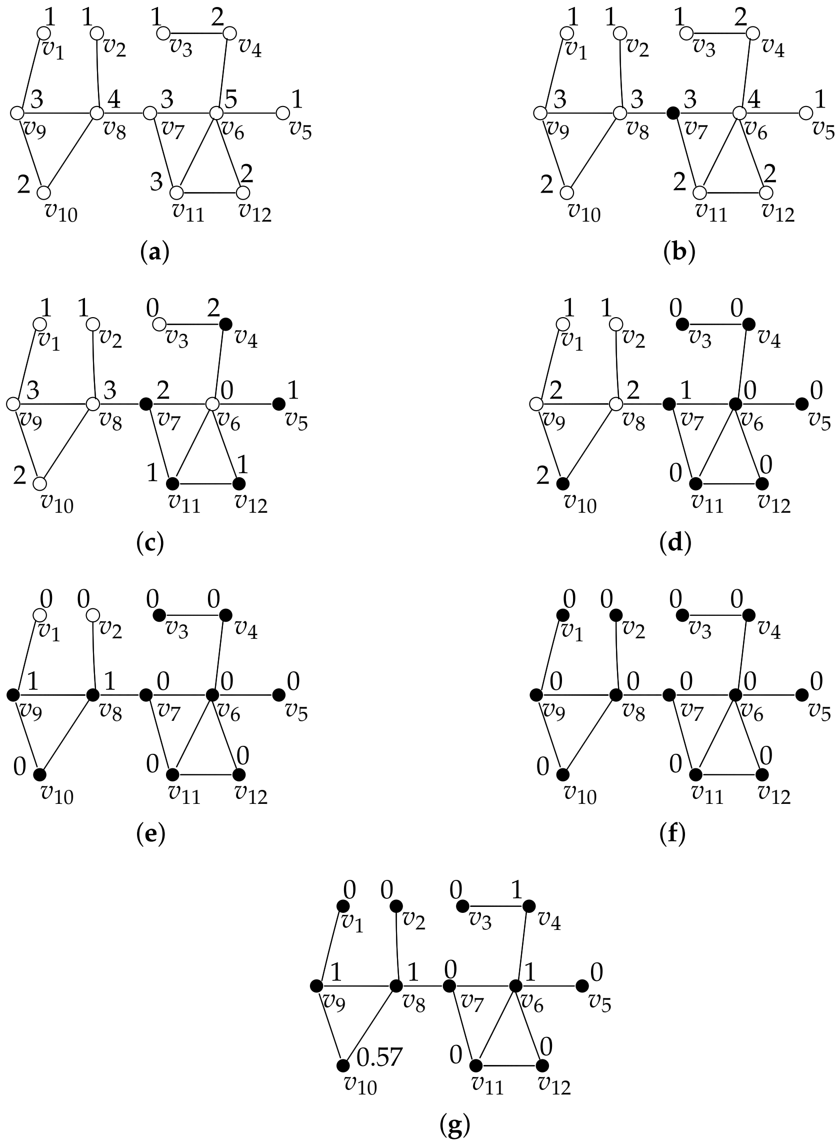

Figure 1. An illustration of Algorithm 1, in which $k = 5$ and $\Delta = 5$. (**a**) is the topology graph, $\tilde{\delta}(v_i)$ of each vertex is marked on the graph. (**b**) When $\hbar = k - 1 = 4$ and $m = 4$, v_7 is colored black and we update $\tilde{\delta}(v_i)$. (**c**) When $\hbar = k - 1 = 4$ and $m : 3 \to 0$, v_4, v_5, v_{11}, v_{12} are colored black and we update $\tilde{\delta}(v_i)$. (**d**) When $\hbar = k - 2 = 3$ and $m : 4 \to 0$, v_3, v_6, v_{10} are colored black and we update $\tilde{\delta}(v_i)$. (**e**) When $\hbar = k - 3 = 2$ and $m : 4 \to 0$, v_8, v_9 are colored black and we update $\tilde{\delta}(v_i)$. (**f**) When $\hbar = k - 4 = 1$ and $m : 4 \to 0$, v_1, v_2 are colored black and we update $\tilde{\delta}(v_i)$. By executing the algorithm, the final x_i of all vertices is shown in (**g**).

Based on the induction hypothesis, we can obtain, for each vertex v_i, $\tilde{\delta}(v_i) \le \Delta^{\frac{\hbar+1}{k}}$ in each outer loop iteration start. As $\tau^{(2)}(v_i)$ is the maximum dynamic degree in the second neighborhood of v_i, we can obtain $\tau^{(2)}(v_i) \le \Delta^{\frac{\hbar+1}{k}}$ for each vertex v_i. Hence, we have

$$\tau^{(2)}(v_i)^{\frac{\hbar}{\hbar+1}} \le \Delta^{\frac{\hbar+1}{k} \cdot \frac{\hbar}{\hbar+1}} = \Delta^{\frac{\hbar}{k}}.$$

We complete the proof. $\square$

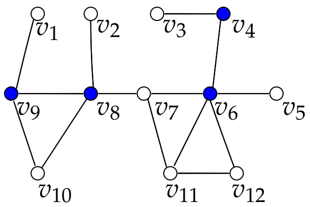

Figure 2. The optimal total dominating set, in which $k = 5$ and $\Delta = 5$.

Table 1. Initial and new values of some variables.

		v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}
	x_i	0	0	0	0	0	0	0	0	0	0	0	0
initial	$\tilde{\delta}(v_i)$	1	1	1	2	1	5	3	4	3	2	3	2
	$\tau^{(2)}(v_i)$	4	4	5	5	5	4	5	5	4	4	5	5
	$\tau^{(2)}(v_i)^{\frac{4}{5}}$	3.03	3.03	3.62	3.62	3.62	3.03	3.62	3.62	3.03	3.03	3.62	3.62
$\hbar = 4$	$a(v_i)$	0	1	0	1	1	0	2	0	1	1	1	1
	$a^{(1)}(v_i)$	1	0	1	0	0	2	1	2	1	1	2	1
$m = 4$	x_i	0	0	0	0	0	0.57	0	0.57	0	0	0	0
	$\tilde{\delta}(v_i)$	1	1	1	2	1	4	3	3	3	2	2	2
	$a(v_i)$	0	0	0	1	1	0	0	0	0	0	1	1
$\hbar = 4$	$a^{(1)}(v_i)$	0	0	1	0	0	1	1	0	0	0	1	1
	x_i	0	0	0	0	0	1	0	0.57	0	0	0	0
$m = 3$	$\tilde{\delta}(v_i)$	1	1	0	2	1	0	2	3	3	2	1	1
$2 \xrightarrow{m} 0$	$\tau^{(2)}(v_i)$	3	3	2	2	2	3	3	3	3	3	3	2

Lemma 2. *Before assigning a new x_i to v_i, for each vertex $v_i \in V$, we can obtain*

$$a(v_i) \le \Delta^{\frac{m+1}{k}}.$$

Proof of Lemma 2. We also prove it by using induction.

Firstly, when $m = k - 1$, $\Delta^{\frac{m+1}{k}} = \Delta$. According to the definitions of $a(v_i)$ and Δ, it is clear that $a(v_i) \le \Delta$.

Secondly, when $m \ne k - 1$, we need to prove that all vertices v_i with $a(v_i) > \Delta^{\frac{m}{k}}$ are colored black when the inner loop iteration ends. We apply the induction hypothesis to the other inner loop iterations, so we can obtain $a(v_i) \le \Delta^{\frac{m+1}{k}}$ for each vertex v_i. Hence, the x_j of each active vertex v_j is assigned in line 17 as

$$x_j \ge a^{(1)}(v_j)^{-\frac{m}{m+1}} \ge \frac{1}{\Delta^{\frac{m+1}{k} \cdot \frac{m}{m+1}}} = \frac{1}{\Delta^{\frac{m}{k}}}.$$

If each vertex v_i has more than $\Delta^{\frac{m}{k}}$ active vertices in $N(v_i)$, v_i will be covered. So, the conclusion holds. $\square$

Then, we analyze all the increased x_i in the inner loop iterations. It is difficult to directly compute the sum of all the increased x_i. Therefore, for each vertex v_i, we introduce a new variable z_i. All x_i is initialized as 0 in line 4 of Algorithm 1. When the vertex v_i increases the x_i over time, the increased x_i is equally distributed to the z_j of the vertices in $\{v_j \in N(v_i) \mid v_j$ which is colored white before the increment of x_i at line 17$\}$. Hence, we can obtain that $\sum z_j$ is equal to the sum of all the increased x_i in each outer loop iteration. So, we only need to show that all z_i are bounded, as then all the increased x_i are also bounded.

Lemma 3. *For each vertex $v_i \in V$, we can obtain $z_i \leq \frac{1+\Delta^{1/k}}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}}$ when each outer loop iteration ends.*

Proof of Lemma 3. Due to all $z_i = 0$ in line 4, we only need to analyze a single outer loop iteration $\hbar$. From the execution of the algorithm, we know that all x_i are increased in line 17. Therefore, z_i can only be increased there if the vertex v_i is white. For each white vertex v_i, we need to discuss two cases during the current outer loop $\hbar$. The first case is that, when all inner loop iterations end, v_i is still white. The second case is that, when the remaining inner loop iterations end, the vertex v_i is or becomes black.

For the first case, we can easily obtain $\sum_{v_j \in N(v_i)} x_j \leq 1$. Since all the increased x are distributed among at least $\tau^{(2)}(v_j)^{\frac{\hbar}{\hbar+1}}$ z_i, we can obtain

$$
\begin{aligned}
z_i &\leq \sum_{v_j \in N(v_i)} \frac{x_j}{\tau^{(2)}(v_j)^{\frac{\hbar}{\hbar+1}}} \\
&\leq \frac{1}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}},
\end{aligned} \tag{1}
$$

where the second inequality holds because $\tau^{(1)}(v_i)$ and $\tau^{(2)}(v_j)$ denote the maximum dynamic degree $\tilde{\delta}(v_i)$ in $N(v_i)$ and the second neighborhood of v_i, respectively, so we have $\tau^{(2)}(v_j) \geq \tau^{(1)}(v_i)$.

In the second case, we can see that only white vertices can increase their z. Since the $\tilde{\delta}(v_i)$ will become smaller over time, these active vertices $v_j \in N(v_i)$ become active in the previous iterations. Hence, we can see that each active vertex x_j contributes to the z_i at most

$$
\frac{x_j}{\tilde{\delta}(v_i)} \leq \frac{1}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}} \cdot a^{(1)}(v_j)^{-\frac{m}{m+1}}.
$$

Due to v_i having $a(v_i)$ active vertices in $N(v_i)$, the upper bound of the increased z_i is

$$
\begin{aligned}
\frac{1}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}} \cdot \frac{1}{a^{(1)}(v_j)^{\frac{m}{m+1}}} \cdot a(v_i) &\leq \frac{1}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}} \cdot \frac{1}{a(v_i)^{\frac{m}{m+1}}} \cdot a(v_i) \\
&= \frac{a(v_i)^{\frac{1}{m+1}}}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}},
\end{aligned} \tag{2}
$$

where the first inequality follows from $a(v_i) \leq a^{(1)}(v_j)$.

Combining these results with Lemma 2, by adding (1) and (2), we can obtain

$$
\begin{aligned}
z_i &\le \frac{1}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}} + \frac{a(v_i)^{\frac{1}{m+1}}}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}} \\
&\le \frac{1 + (\Delta^{\frac{m+1}{k}})^{\frac{1}{m+1}}}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}} \\
&= \frac{1 + \Delta^{1/k}}{\tau^{(1)}(v_i)^{\frac{\hbar}{\hbar+1}}}.
\end{aligned}
$$

We complete the proof. □

Based on the two lemmas, we will analyze the correctness, approximation ratio, and time complexity of Algorithm 1 in the following. Firstly, we show that the algorithm outputs a feasible solution to the linear programming relaxation of the minimum total dominating set problem.

Theorem 1. *The output result $\bar{x}$ of Algorithm 1 is a feasible solution to linear programming relaxation of minimum total dominating set problem LP_{MTDS}.*

Proof of Theorem 1. According to the execution of Algorithm 1, v_i will be colored 'black' only when the vertex v_i satifies $\sum_{v_j \in N(v_i)} x_j \ge 1$, i.e., x_i belongs to the feasible solution of LP_{MTDS}. For each vertex v_i, we can obtain $\tilde{\delta}(v_i) = 0$ and $\tau^{(2)}(v_i) = 0$ at the end of the iteration ($\hbar = 0, m = 0$), so the algorithm will terminate and all vertices are colored 'black'. Hence, the output result $\bar{x}$ of Algorithm 1 is a feasible solution for LP_{MTDS}. □

Secondly, we analyze the approximation ratio of Algorithm 1.

Theorem 2. *For a given network graph G and a positive integer k, Algorithm 1 is a $k(1 + \Delta^{\frac{1}{k}})\Delta^{\frac{1}{k}}$-approximation for the minimum total dominating set problem in G.*

Proof of Theorem 2. According to the above description of z_j, $\sum z_j$ is equal to the sum of all the increased x_i in each outer loop iteration of Algorithm 1. From Lemma 3, we know that the z_j of all vertices satisfy $z_j \le \frac{1+\Delta^{1/k}}{\tau^{(1)}(v_j)^{\frac{\hbar}{\hbar+1}}}$ when each outer loop iteration ends. So, for each vertex v_i, the sum of the z_j in $N(v_i)$ of the vertex v_i in the outer loop iteration is at most

$$
\begin{aligned}
\sum_{v_j \in N(v_i)} z_j &\le \frac{1 + \Delta^{1/k}}{\tau^{(1)}(v_j)^{\frac{\hbar}{\hbar+1}}} \cdot \tilde{\delta}(v_i) \\
&\le \frac{1 + \Delta^{1/k}}{\tilde{\delta}(v_i)^{\frac{\hbar}{\hbar+1}}} \cdot \tilde{\delta}(v_i) \\
&= (1 + \Delta^{1/k}) \cdot \tilde{\delta}(v_i)^{\frac{1}{\hbar+1}} \\
&\le (1 + \Delta^{\frac{1}{k}}) \cdot \Delta^{\frac{1}{k}},
\end{aligned}
$$

where the second inequality holds because $\tau^{(1)}(v_j)$ is the maximum dynamic degree $\tilde{\delta}(v_i)$ in $N(v_j)$, $v_i \in N(v_j)$, hence $\tau^{(1)}(v_j) \ge \tilde{\delta}(v_i)$. The fourth inequality follows from Lemma 1, $\tilde{\delta}(v_i) \le \Delta^{\frac{\hbar+1}{k}}$.

Next, for each vertex v_j, if we let $y_j := \frac{z_j}{(1+\Delta^{\frac{1}{k}})\Delta^{\frac{1}{k}}}$, then we can see that $\sum_{v_j \in N(v_i)} y_j \le 1$ holds for each vertex $v_i \in V$; hence, the $\bar{y}$ is a feasible solution of DLP_{MTDS}. According to the weak duality theorem of linear programming, we can see that $\sum y_j$ is a lower bound of an optimal TDS (TDS_{OPT}). Therefore, for each outer loop iteration, we can obtain

$$\sum_{j=1}^{n} z_j \leq (1+\Delta^{\frac{1}{k}}) \cdot \Delta^{\frac{1}{k}} \cdot |TDS_{OPT}|.$$

On the other hand, there are k iterations in the outer loop. So, we can obtain

$$\sum_{i=1}^{n} x_i \leq k(1+\Delta^{\frac{1}{k}}) \cdot \Delta^{\frac{1}{k}} \cdot |TDS_{OPT}|.$$

We complete the proof. □

Finally, we analyze the running time of Algorithm 1.

Theorem 3. *The number of communication rounds for Algorithm 1 is* $4k^2 + O(k)$.

Proof of Theorem 3. Firstly, it can be seen from the execution of the algorithm that every inner loop iteration needs to send four messages to all its neighbors in the algorithm, and there are k inner loop iterations, so it takes $4k$ communication rounds. Secondly, every outer loop iteration needs to send two messages to all neighbors in the algorithm; hence, it takes $4k+2$ communication rounds for each outer loop iteration. There are k outer loop iterations, so it takes $k(4k+2)$ communication rounds for all outer loop iterations. Finally, we need to calculate some values before the outer loop iteration starts, which takes $O(k)$ communication rounds. Hence, we can see that the number of communication rounds in the algorithm is $k(4k+2) + O(k) = 4k^2 + O(k)$. □

5. Conclusions

In this paper, we present a distributed approximation algorithm for the total dominating set problem by using the maximum dynamic degree in the second neighborhood of the vertex and LP relaxation techniques. Our algorithm obtained only a relaxed solution; further considerations can be given on how to design a random rounding algorithm to obtain an integer TDS from the relaxed solution. It is interesting to consider this problem in wireless networks so that we can take advantage of the specificity of wireless networks to improve the performance of the algorithm. In practical applications, if the vertices in a graph are weighted, additional considerations could include how to design an approximation algorithm to solve the minimum total dominating set problem.

Author Contributions: Conceptualization, L.W.; methodology, L.W.; validation, L.W. and W.W.; formal analysis, L.W. and W.W.; investigation, L.W. and W.W.; writing—review and editing, L.W.; project administration, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: The first author is supported by NSFC (No. 62272215).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: This manuscript has no associated data.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LP	Linear programming
TDS	Total dominating set
MTDS	Minimum total dominating set
ILP_{MTDS}	Integer linear programming of minimum total dominating set problem
LP_{MTDS}	Linear programming relaxation of minimum total dominating set problem
DLP_{MTDS}	Dual linear programming of minimum total dominating set problem
TDS_{OPT}	Optimal total dominating set

Appendix A

Algorithm A1 Approximating LP_{MTDS}

Input: $G = (V, E)$, a positive integer k and maximum degree Δ.
Output: A feasible solution $\overline{x}$ for LP_{MTDS}.

```
initially x_i := 0, δ̃(v_i) := δ(v_i);
for ħ := k − 1 to 0 by −1 do
    set z_i := 0; a(v_i) := 0
    for m := k − 1 to 0 by −1 do
        send the color of node v_i to all nodes in N(v_i);
        update δ̃(v_i) := |{v_j ∈ N(v_i)|the node v_j is white}|;
        if δ̃(v_i) ≥ Δ^(ħ/k) then
            x_i := max{x_i, 1/Δ^(m/k)};
        end if
        send x_i to all nodes in N(v_i);
        update the color of node v_i and the values z_i, a(v_i);
    end for
end for
```

References

1. Jiang, P.; Liu, J.; Wu, F.; Wang, J.; Xue, A. Node deployment algorithm for underwater sensor networks based on connected dominating set. *Sensors* **2016**, *16*, 388. [CrossRef] [PubMed]
2. Chen, J.; He, K.; Du, R.; Zheng, M.; Xiang, Y.; Yuan, Q. Dominating set and network coding-based routing in wireless mesh networks. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *26*, 423–433. [CrossRef]
3. Fellows, M.R.; Koblitz, N. Combinatorially based cryptography for children (and adults). *Congr. Numer.* **1994**, *99*, 9–41.
4. Kwon, S.; Kang, J.S.; Yeom, Y. Analysis of public-key cryptography using a 3-regular graph with a perfect dominating set. In Proceedings of the IEEE Region 10 Symposium (TENSYMP), Grand Hyatt Jeju, Republic of Korea, 23–25 August 2021; pp. 1–6.
5. Allesina, S.; Bodini, A. Who dominates whom in the ecosystem? Energy flow bottlenecks and cascading extinctions. *J. Theor. Biol.* **2004**, *230*, 351–358. [CrossRef]
6. Haynes, T.W.; Hedetniemi, S.M.; Hedetniemi, S.T.; Henning, M.A. Domination in graphs applied to electric power networks. *SIAM J. Discret. Math.* **2002**, *15*, 519–529. [CrossRef]
7. Haynes, T.W.; Hedetniemi, S.; Slater, P. *Fundamentals of Domination in Graphs*; CRC Press: Boca Raton, FL, USA, 2013.
8. Haynes, T.W. *Domination in Graphs: Volume 2: Advanced Topics*; Routledge: London, UK, 2017.
9. Haynes, T.W.; Hedetniemi, S.; Henning, M.A. *Topics in Domination in Graphs*; Springer Nature: Berlin/Heidelberg, Germany, 2020.
10. Haynes, T.W.; Hedetniemi, S.; Henning, M.A. *Structures of Domination in Graphs*; Springer: Cham, Switzerland, 2021.
11. Alzoubi, K.M.; Wan, P.J.; Frieder, O. Maximal independent set, weakly-connected dominating set, and induced spanners in wireless ad hoc networks. *Int. J. Found. Comput. Sci.* **2003**, *14*, 287–303. [CrossRef]
12. Das, B.; Bharghavan, V. Routing in ad-hoc networks using minimum connected dominating sets. In Proceedings of the ICC'97-International Conference on Communications, Montreal, QC, Canada, 8–12 June 1997; pp. 376–380.
13. Belhoul, Y.; Yahiaoui, S.; Kheddouci, H. Efficient self-stabilizing algorithms for minimal total k-dominating sets in graphs. *Inf. Process. Lett.* **2014**, *114*, 339–343. [CrossRef]
14. Cockayne, E.J.; Dawes, R.M.; Hedetniemi, S.T. Total domination in graphs. *Networks* **1980**, *10*, 211–219. [CrossRef]
15. Awerbuch, B.; Goldberg, A.V.; Luby, M.; Plotkin, S.A. Network decomposition and locality in distributed computation. In Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, Research Triangle Park, NC, USA, 30 October–1 November 1989; pp. 364–369.
16. Alipour, S.; Futuhi, E.; Karimi, S. On distributed algorithms for minimum dominating set problem, from theory to application. *arXiv* **2020**, arXiv:2012.04883. [CrossRef]

17. Alipour, S.; Jafari, A. A local constant approximation factor algorithm for minimum dominating set of certain planar graphs. In Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures, Virtual, 15–17 July 2020; pp. 501–502.
18. Haynes, T.W.; Hedetniemi, S.T.; Slater, P.J. *Fundamentals of Domination in Graphs*; Chapman and Hall/CRC: New York, NY, USA, 1998.
19. Haynes, T.W.; Hedetniemi, S.T.; Slater, P.J. *Domination in Graphs: Advanced Topics*; Chapman and Hall/CRC Pure and Applied Mathematics Series; Marcel Dekker: New York, NY, USA, 1998; Volume 209.
20. Henning, M.A. A survey of selected recent results on total domination in graphs. *Discret. Math.* **2009**, *309*, 32–63. [CrossRef]
21. Henning, M.A.; Yeo, A. *Total Domination in Graphs*; Springer: New York, NY, USA, 2013.
22. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; Freeman: San Francisco, CA, USA, 1978.
23. Khanna, S.; Motwani, R.; Sudan, M.; Vazirani, U. On syntactic versus computational views of approximability. *SIAM J. Comput.* **1998**, *28*, 164–191. [CrossRef]
24. Papademitriou, C.; Yannakakis, M. Optimization, approximation and complexity classes. In Proceeding of the 20th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, 2–4 May 1988; pp. 229–234.
25. Crescenzi, P.; Kann, V. *A Compendium of NP Optimization Problems*; Technical Report SI/RR-95/02; Department of Computer Science, University of Rome "La Sapienza": Rome, Italy, 1995.
26. Laskar, R.; Pfaff, J.; Hedetniemi, S.M.; Hedetniemi, S.T. On the algorithmic complexity of total domination. *SIAM J. Algebr. Discret. Methods* **1984**, *5*, 420–425. [CrossRef]
27. Henning, M.A.; Yeo, A. A transition from total domination in graphs to transversals in hypergraphs. *Quaest. Math.* **2007**, *30*, 417–436. [CrossRef]
28. Chlebík, M.; Chlebíková, J. Approximation hardness of dominating set problems in bounded degree graphs. *Inf. Comput.* **2008**, *206*, 1264–1275. [CrossRef]
29. Zhu, J. Approximation for minimum total dominating set. In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, Seoul, Republic of Korea, 24–26 November 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 119–124.
30. Schaudt, O.; Schrader, R. The complexity of connected dominating sets and total dominating sets with specified induced subgraphs. *Inf. Process. Lett.* **2012**, *112*, 953–957. [CrossRef]
31. Yuan, F.; Li, C.; Gao, X.; Yin, M.; Wang, Y. A novel hybrid algorithm for minimum total dominating set problem. *Mathematics* **2019**, *7*, 222. [CrossRef]
32. Bahadir, S. An algorithm to check the equality of total domination number and double of domination number in graphs. *Turk. J. Math.* **2020**, *44*, 1701–1707. [CrossRef]
33. Hu, S.; Liu, H.; Wang, Y.; Li, R.; Yin, M.; Yang, N. Towards efficient local search for the minimum total dominating set problem. *Appl. Intell.* **2021**, *51*, 8753–8767. [CrossRef]
34. Jena, S.K.; Das, G.K. Total domination in geometric unit disk graphs. In Proceeding of the 33rd Canadian Conference on Computational Geometry (CCCG), Halifax, NS, Canada, 10–12 August 2021; pp. 219–227.
35. Hochbaum, D.S.; Maass, W. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM (JACM)* **1985**, *32*, 130–136. [CrossRef]
36. Goddard, W.; Hedetniemi, S.T.; Jacobs, D.P.; Srimani, P.K. A self-stabilizing distributed algorithm for minimal total domination in an arbitrary system graph. In Proceedings of the International Parallel and Distributed Processing Symposium, Nice, France, 22–26 April 2003.
37. Elkin, M. Distributed approximation: A survey. *ACM SIGACT News* **2004**, *35*, 40–57. [CrossRef]
38. Kuhn, F.; Wattenhofer, R. Constant-time distributed dominating set approximation. *Distrib. Comput.* **2005**, *17*, 303–310. [CrossRef]
39. Jia, L.; Rajaraman, R.; Suel, T. An efficient distributed algorithm for constructing small dominating sets. *Distrib. Comput.* **2002**, *15*, 193–205. [CrossRef]
40. Guha, S.; Khuller, S. Approximation algorithms for connected dominating sets. *Algorithmica* **1998**, *20*, 374–387. [CrossRef]